Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/140604

A Pseudo-Random Number Generator Using Double Pendulum

*Chokri Nouar *and Zine El Abidine Guennoun*

Department of Mathematics, Mohamed V University in Rabat No. 4, Avenue Ibn Battouta B. P. 1014 RP, Rabat, Morocco

Received: 15 Oct. 2019, Revised: 3 Apr. 2020, Accepted: 6 May. 2020
Published online: 1 Nov. 2020

Abstract: Chaos in the double pendulum motion has been proved in several studies. Despite this useful cryptographic propriety, this system has not been applied to cryptography yet. This paper presents a new pseudo random number generator based on a double pendulum. Randomness of the numbers generated by the proposed generator is successfully tested by NIST and DIEHARDER tests. The results of the security analysis asserted the appropriateness of the new generator for cryptographic applications.

Keywords: Chaotic systems, DIEHARDER, Double Pendulum, NIST, Pseudo-Random Number Generator.

1 Introduction

Recently, the pseudo random number generators have become a ubiquitous tool used in numerous areas such as numerical analysis, statistical sampling, gaming industry, computer simulations, computer security (keygen, captacha...), cryptography , ... etc. [1].

Chaotic dynamical systems are highly sensitive to initial conditions and parameters [2]. This propriety makes the pseudo random numbers generators based on them appropriate for encryption algorithms. The idea of designing a pseudo random numbers generator using chaotic dynamical systems was proposed by Oishi and Inoue in 1982 . Several pseudo random number generators were suggested in their paper [3]. The double pendulum is a dynamic chaotic system [4] that consists of two-point masses at the end of light rods. It is a simple physical system that exhibits a strong sensitivity to initial conditions [5].

The motion of a double pendulum is given by a set of ordinary differential equations [6]. First, we change the continuous equations into discrete counterparts using the Euler method with very small steps. Second, we collect the Cartesian coordinates of the second mass; the couples (x, y) of every moment. Then, we extract the numbers starting form the fourth digit after the decimal point.

This paper presents a pseudo random number generator based on a double pendulum. The produced sequences were subjected to an experimental study to test randomness and the chaotic behavior of the generator. The sequences stream has successfully passed various statistical tests, and the generator is highly sensitive to one bit change in the keys.

The rest of this paper is organized as follows. In section 2, the double pendulum and its motion equations are introduced. In section 3, a detailed description of our PRNG is presented. Section 4 is dedicated to the statistical analysis and validation of the number sequences generated by our generator.

2 Double Pendulum

A double pendulum consists of a mass $m1$, attached by a massless rod of length $l1$ and a mass $m2$ attached at the mass $m1$ by another massless rod of length $l2$. The system freely rotates in a vertical plane. This means that the first pendulum is attached freely at the second but both are constrained to oscillate in the same plane. See Figure 1.

The double pendulum is very sensitive to initial conditions and its motion exhibits chaotic behavior [4].

* Corresponding author e-mail: chokri.nouar@gmail.com

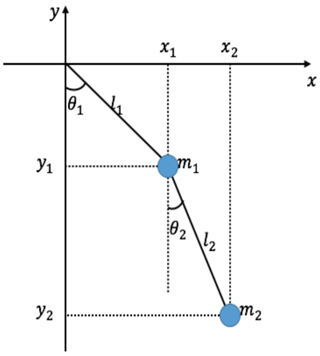

Fig. 1: Double pendulum

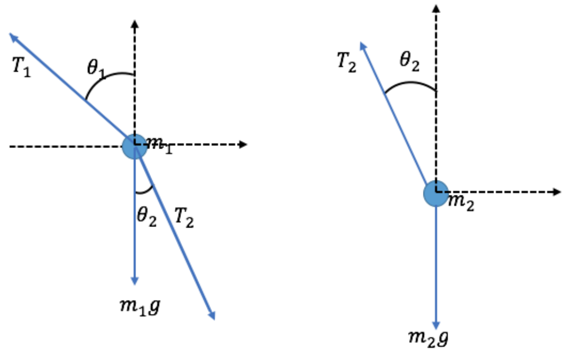

Fig. 2: The forces applied to m1 and m2

2.1 The motion equations

Equations of motion are usually deduced using the Lagrangian, Hamiltonian or Newtonian methods. In this paper, we use the Newtonian method.

The position equations are given by the relations below

$$\begin{cases} x_1 = l_1 sin(\theta_1) \\ y_1 = -l_1 cos(\theta_1) \end{cases}$$
$$\begin{cases} x_2 = x_1 + l_2 sin(\theta_2) \\ y_2 = y_1 - l_2 cos(\theta_2) \end{cases}$$

Speed is the derivative of the position with respect to the time ($v = \dot{x}$).

$$\begin{cases} \dot{x}_1 = \dot{\theta}_1 l_1 cos(\theta_1) \\ \dot{y}_1 = \dot{\theta}_1 l_1 sin(\theta_1) \end{cases}$$
$$\begin{cases} \dot{x}_2 = \dot{x}_1 + \dot{\theta}_2 l_2 cos(\theta_2) \\ \dot{y}_2 = \dot{y}_1 + \dot{\theta}_2 l_2 sin(\theta_2) \end{cases}$$

and the acceleration is the second derivative ($a = \dot{v} = \ddot{x}$), so

$$\begin{cases} \ddot{x}_1 = -\dot{\theta}_1^2 l_1 sin(\theta_1) + \ddot{\theta}_1 l_1 cos(\theta_1) \\ \ddot{y}_1 = \dot{\theta}_1^2 l_1 cos(\theta_1) + \ddot{\theta}_1 l_1 sin(\theta_1) \end{cases}$$
$$\begin{cases} \ddot{x}_2 = \ddot{x}_1 - \dot{\theta}_2^2 l_2 sin(\theta_2) + \ddot{\theta}_2 l_2 cos(\theta_2) \\ \ddot{y}_2 = \ddot{y}_1 + \dot{\theta}_2^2 l_2 cos(\theta_2) + \ddot{\theta}_2 l_2 sin(\theta_2) \end{cases}$$

We shall denote by :
T_1 : the tension of the first rod,
T_2 : the tension of the second rod,
m_1 : the mass of $m1$,
m_2 : the mass of $m2$,
g : the gravitational constant.

Applying the second Newton's law : $\sum F = ma$
to the first mass we have :

$$m_1 \ddot{x}_1 = -T_1 sin(\theta_1) + T_2 sin(\theta_2) \quad (1)$$
$$m_1 \ddot{y}_1 = T_1 cos(\theta_1) - T_2 cos(\theta_2) - m_1 g \quad (2)$$

and when applied to the second mass we have :

$$m_2 \ddot{x}_2 = -T_2 sin(\theta_2) \quad (3)$$
$$m_2 \ddot{y}_2 = T_2 cos(\theta_2) - m_2 g \quad (4)$$

Applying some algebraic manipulations to our equations, we find the expressions of $\ddot{\theta}_1$ and $\ddot{\theta}_2$.

From (1) and (3):

$$m_1 \ddot{x}_1 = -T_1 sin(\theta_1) - m_2 \ddot{x}_2$$
$$\Rightarrow \quad m_1 \ddot{x}_1 + m_2 \ddot{x}_2 = -T_1 sin(\theta_1) \quad (5)$$

from (2) and (4):

$$m_1 \ddot{y}_1 = T_1 cos(\theta_1) - m_2 \ddot{y}_2 - m_2 g - m_1 g$$
$$\Rightarrow \quad m_1 \ddot{y}_1 + m_2 \ddot{y}_2 + m_2 g + m_1 g = T_1 cos(\theta_1) \quad (6)$$

Now from (5) and (6), we have the result (7):

$$sin(\theta_1)(m_1 \ddot{y}_1 + m_2 \ddot{y}_2 + m_2 g + m_1 g) = -cos(\theta_1) m_1 \ddot{x}_1 - cos(\theta_1) m_2 \ddot{x}_2 \quad (7)$$

and from (3) and (4) we have the result (8)

$$sin(\theta_2)(m_2 \ddot{y}_2 + m_2 g) = -cos(\theta_2)(m_2 \ddot{x}_2) \quad (8)$$

Finally, we replace $\ddot{x}$ and $\ddot{y}$ in results (7) and (8) with the equations of acceleration. After calculations and some simplifications, the motion equations of the double pendulum are given by the following relations:

$$\ddot{\theta}_1 = \frac{-g(2m_1 + m_2) sin\theta_1 - m_2 g sin(\theta_1 - 2\theta_2)}{l_1(2m_1 + m_2 - m_2 cos(2\theta_1 - 2\theta_2))} - \frac{2 sin(\theta_1 - \theta_2) m2 (\dot{\theta}_2^2 l_2 + \dot{\theta}_1^2 l_1 cos(\theta_1 - \theta_2))}{l_1(2m_1 + m_2 - m_2 cos(2\theta_1 - 2\theta_2))} \quad (9)$$

$$\ddot{\theta}_2 = \frac{2 sin(\theta_1 - \theta_2)[\dot{\theta}_1^2 l_1 (m_1 + m_2)}{l_2(2m_1 + m_2 - m_2 cos(2\theta_1 - 2\theta_2))} + \frac{g(m_1 + m_2) cos(\theta_1) + \dot{\theta}_2^2 l_2 m_2 cos(\theta_1 - \theta_2)]}{l_2(2m_1 + m_2 - m_2 cos(2\theta_1 - 2\theta_2))} \quad (10)$$

2.2 Euler method

The explicit Euler method is a general principle that allows to discretize first degree and first order differential equations with a given initial value. When the step size minimizes, accuracy of the Euler method maximizes.

Given an initial value problem :

$$\dot{y}(t) = f(t, y(t)), \qquad y(t_0) = y_0$$

Let h be the step size, so $t_{n+1} = t_n + h, \quad n \in \mathbb{N}$, and the differential equation is, as follows:

$$y_{n+1} = y_n + hf(t_n, y_n), \quad y(t_n) = y_n$$

2.3 The positions suite

The present paper focuses on the Cartesian coordinates of the second mass $(x_2; y_2)$, given that this position depends on the angles θ_1 and θ_2 by the following relations :

$$\begin{cases} x_2 = l_1 sin(\theta_1) + l_2 sin(\theta_2) \\ y_2 = -l_1 cos(\theta_1) - l_2 cos(\theta_2) \end{cases}$$

From the acceleration $\ddot{\theta}_1$ and $\ddot{\theta}_2$ we use the explicit Euler method to define the speed $\dot{\theta}_1$ and $\dot{\theta}_2$, as well as the position θ_1 and θ_2.

We have :

$$\dot{\theta}_{1,n+1} = \dot{\theta}_{1,n} + h\ddot{\theta}_{1,n} \quad and \quad \dot{\theta}_{2,n+1} = \dot{\theta}_{2,n} + h\ddot{\theta}_{2,n}$$

likewise :

$$\theta_{1,n+1} = \theta_{1,n} + h\dot{\theta}_{1,n} \quad and \quad \theta_{2,n+1} = \theta_{2,n} + h\dot{\theta}_{2,n}$$

Hence, we have the positions suite, as follows:

$$\begin{cases} x_{2,n} = l_1 sin(\theta_{1,n}) + l_2 sin(\theta_{2,n}) \\ y_{2,n} = -l_1 cos(\theta_{1,n}) - l_2 cos(\theta_{2,n}) \end{cases}$$

3 The Proposed Generator

This paper presents a new pseudo-random number generator based on a double pendulum; it is a deterministic generator initialized by a key K of more than four characters size whose output is a cryptographically-secured binary sequence.

Considering a double pendulum with the parameters $l_1, l_2, m_1, m_2, \theta_1, \theta_2$, each one is calculated from the key K using a new method based on XOR and permutation of the bits derived by ASCII representation of the K.

After the initialization of the parameters $l_1, l_2, m_1, m_2, \theta_1, \theta_2$, our generator starts producing the numbers needed to construct the final sequence $S = S_1 ... S_n$ with $S_i = X_i \| Y_i$, where X_i and Y_i are two 32-bit numbers generated in the i^{th} step. "$\|$" represents concatenation between tow bits.

3.1 The calculation of the initialization values

The initial conditions $l_1, l_2, m_1, m_2, \theta_1, \theta_2$ are calculated from a binary string of any length $n \geq 32bits$ which represents the key $K = (k_1 k_2 k_3 ... k_n)_2$. Hence we extract 64 bits for each parameter value from the K by a method based on a new technique of initialization.

In the first step, the key binaries are divided into four parts (i.e. k_1, k_2, k_3 and k_4) and from these four parts all parameters are built by XOR and concatenation. This operation stops when we get 64 bits for each part.

$$\begin{aligned} L_1 &= k_1 \oplus k_2 \| k_1 \oplus k_2 \| ... \\ L_2 &= k_1 \oplus k_3 \| k_1 \oplus k_3 \| ... \\ M_1 &= k_1 \oplus k_4 \| k_1 \oplus k_4 \| ... \\ M_2 &= k_2 \oplus k_3 \| k_2 \oplus k_3 \| ... \\ T_1 &= k_2 \oplus k_4 \| k_2 \oplus k_4 \| ... \\ T_2 &= k_3 \oplus k_4 \| k_3 \oplus k_4 \| ... \\ R &= k_1 \oplus k_2 \| k_3 \oplus k_4 \| ... \end{aligned}$$

In the second step, all parameters are represented in the binary form as follows

$$\begin{aligned} L_1 &= (k_0^{L_1} k_1^{L_1} k_2^{L_1} k_3^{L_1} ... k_{60}^{L_1} k_{61}^{L_1} k_{62}^{L_1} k_{63}^{L_1}) \\ L_2 &= (k_0^{L_2} k_1^{L_2} k_2^{L_2} k_3^{L_2} ... k_{60}^{L_2} k_{61}^{L_2} k_{62}^{L_2} k_{63}^{L_2}) \\ M_1 &= (k_0^{M_1} k_1^{M_1} k_2^{M_1} k_3^{M_1} ... k_{60}^{M_1} k_{61}^{M_1} k_{62}^{M_1} k_{63}^{M_1}) \\ M_2 &= (k_0^{M_2} k_1^{M_2} k_2^{M_2} k_3^{M_2} ... k_{60}^{M_2} k_{61}^{M_2} k_{62}^{M_2} k_{63}^{M_2}) \\ T_1 &= (k_0^{T_1} k_1^{T_1} k_2^{T_1} k_3^{T_1} ... k_{60}^{T_1} k_{61}^{T_1} k_{62}^{T_1} k_{63}^{T_1}) \\ T_2 &= (k_0^{T_2} k_1^{T_2} k_2^{T_2} k_3^{T_2} ... k_{60}^{T_2} k_{61}^{T_2} k_{62}^{T_2} k_{63}^{T_2}) \\ R &= (k_0^R k_1^R k_2^R k_3^R ... k_{60}^R k_{61}^R k_{62}^R k_{63}^R) \end{aligned}$$

Finally, from this binary representation, we calculate the real values of our parameters by a binary to decimal conversion.

$$l_1 = \sum_{i=0}^{63} \frac{k_i^{L_1} \times 2^{63-i}}{2^{63}} \qquad l_2 = \sum_{i=0}^{63} \frac{k_i^{L_2} \times 2^{63-i}}{2^{63}}$$

$$m_1 = \sum_{i=0}^{63} \frac{k_i^{M_1} \times 2^{63-i}}{2^{63}} \qquad m_2 = \sum_{i=0}^{63} \frac{k_i^{M_2} \times 2^{63-i}}{2^{63}}$$

$$\theta_1 = \sum_{i=0}^{63} \frac{k_i^{T_1} \times 2^{63-i}}{2^{63}} \qquad \theta_2 = \sum_{i=0}^{63} \frac{k_i^{T_2} \times 2^{63-i}}{2^{63}}$$

$$r = \sum_{i=0}^{63} \frac{k_i^{R} \times 2^{63-i}}{2^{59}}$$

Consequently, all initial conditions values are included in the interval $[0; 2]$ and $r \in [0; 10^4]$. **Algorithm 1** is used to calculate the initial values

3.2 Generating the pseudo-random sequence

After extracting the initial values $l_1, l_2, m_1, m_2, r, \theta_1$ and θ_2, our system gets ready to generate the pseudo random

Algorithm 1 initialization

Input key $K = (k_1k_2...k_n)_2$ a binary string of any length
Output initiation values $l_1, l_2, m_1, m_2, \theta_1$, θ_2 and r
$L \leftarrow Length(K)$
for $i \leftarrow 0$ **to** $L/4$ **do**
  $k_1[i] \leftarrow K[i]$
  $k_2[i] \leftarrow K[L/4+i]$
  $k_3[i] \leftarrow K[L/2+i]$
  $k_4[i] \leftarrow K[3L/4+i]$
end for
$j \leftarrow 0$
while $j < 64$ **do**
  $i \leftarrow 0$
  while $L/4 - i > 0$ **do**
    $L_1[j] \leftarrow k_1[i] \oplus k_2[i]$
    $L_2[j] \leftarrow k_1[i] \oplus k_3[i]$
    $M_1[j] \leftarrow k_1[i] \oplus k_4[i]$
    $M_2[j] \leftarrow k_2[i] \oplus k_3[i]$
    $T_1[j] \leftarrow k_2[i] \oplus k_4[i]$
    $T_2[j] \leftarrow k_3[i] \oplus k_4[i]$
    $R[j] \leftarrow k_1[i]$
    if $j >= 64$ **Exit**
  end while
end while
for $i \leftarrow 0$ **to** 64 **do**
  $l_1 \leftarrow l_1 + L_1[i] \times 2^{63-i}$
  $l_2 \leftarrow l_2 + L_2[i] \times 2^{63-i}$
  $m_1 \leftarrow m_1 + M_1[i] \times 2^{63-i}$
  $m_2 \leftarrow m_2 + M_2[i] \times 2^{63-i}$
  $t_1 \leftarrow t_1 + T_1[i] \times 2^{63-i}$
  $t_2 \leftarrow t_2 + T_2[i] \times 2^{63-i}$
  $r \leftarrow r + R[i] \times 2^{63-i}$
end for
$l_1 \leftarrow \frac{l_1}{2^{63}}$; $l_2 \leftarrow \frac{l_2}{2^{63}}$; $m_1 \leftarrow \frac{m_1}{2^{63}}$; $m_2 \leftarrow \frac{m_2}{2^{63}}$;
  $\theta_1 \leftarrow \frac{t_1}{2^{63}}\pi$; $\theta_2 \leftarrow \frac{t_2}{2^{63}}\pi$; $r \leftarrow \frac{r}{2^{59}}$;
return $l_1; l_2; m_1; m_2; \theta_1; \theta_2;\ r$

number sequences by the following relations.

$$\begin{cases} x_{2,n} = l_1 sin(\theta_{1,n}) + l_2 sin(\theta_{2,n}) \\ y_{2,n} = -l_1 cos(\theta_{1,n}) - l_2 cos(\theta_{2,n}) \end{cases}$$

$$\begin{cases} \theta_{1,n+1} = \theta_{1,n} + h\dot{\theta}_{1,n} \\ \theta_{2,n+1} = \theta_{2,n} + h\dot{\theta}_{2,n} \end{cases} \quad \begin{cases} \dot{\theta}_{1,n+1} = \dot{\theta}_{1,n} + h\ddot{\theta}_{1,n} \\ \dot{\theta}_{2,n+1} = \dot{\theta}_{2,n} + h\ddot{\theta}_{2,n} \end{cases}$$

The **algorithm** 2 generates a pseudo random binary sequence of a given size F in three steps using two inputs: a key K and the integer F.

step 1: The system loops up to t_0 iterations to avoid the harmful effects of transitional procedures [7], where t_0 is determined from the length of K as follows : $t_0 = r \times length(K)$

step 2: Our system starts generating the pseudo random numbers for each $i \geq t_0$. To construct the sub-sequence S_i, we take the pairs (X_i, Y_i) as shown below :

$$X_i = floor[mod(x_{2,i} \times 10^3, 1)] \times 2^{32}$$

$$Y_i = floor[mod(y_{2,i} \times 10^3, 1)] \times 2^{32}$$

The $floor(x)$ rounds each element of x to the nearest integer less than or equal to x. The $mod(x,y)$ returns the remainder after division of x by y, and the X_i and Y_i are two 32-bit numbers generated in the i^{th} step.

step 3: Calculating the $X_i = (k_{31}k_{30}k_{29}...k_1k_0)$ and $Y_i = (k'_{31}k'_{30}k'_{29}...k'_1k'_0)$, we construct the sub sequence S_i by a bit-by-bit concatenation, such as

$$S_i = X_i \| Y_i = (k_{31}k'_{31}k_{30}k'_{30}k_{29}k'_{29}...k_1k'_1k_0k'_0)$$

The final sequence S of our system is the concatenation of the sub-sequences $S = S_1 \| S_2 \| S_3 \| ... \| S_F$

Algorithm 2 Generation

Input key $K = (k_1k_2...k_n)_2$ and F the length of the requested binary sequence
Output S the random binary sequence
$r, l_1, l_2, m_1, m_2, \theta_1, \theta_2 \leftarrow$ Initialize(K)
$t_0 \leftarrow r \times length(K)$
$i, j, k, S \leftarrow 0$
$v_1, v_2 \leftarrow 0$
$h \leftarrow 0.002$
while $j \leq F$ **do**
  $a_1 \leftarrow \ddot{\theta}_1$ and $a_2 \leftarrow \ddot{\theta}_2$ using the motion equations
  if $i \leq t_0$ **then**
    $v_1 \leftarrow v_1 + h * a_1$ and $v_2 \leftarrow v_2 + h * a_2$
    $\theta_1 \leftarrow \theta_1 + h * v_1$ and $\theta_2 \leftarrow \theta_2 + h * v_2$
    $x_2 \leftarrow l_1 * sin(\theta_1) + l_2 * sin(\theta_2)$
    $y_2 \leftarrow -l_1 * cos(\theta_1) - l_2 * cos(\theta_2)$
    $i \leftarrow i + 1$
  else
    $v_1 \leftarrow v_1 + h * a_1$ and $v_2 \leftarrow v_2 + h * a_2$
    $\theta_1 \leftarrow \theta_1 + h * v_1$ and $\theta_2 \leftarrow \theta_2 + h * v_2$
    $x_2 \leftarrow l_1 * sin(\theta_1) + l_2 * sin(\theta_2)$
    $y_2 \leftarrow -l_1 * cos(\theta_1) - l_2 * cos(\theta_2)$
    $X \leftarrow \lfloor (x_2 \times 10^3) mod(1) \rfloor \times 2^{32}$
    $Y \leftarrow \lfloor (y_2 \times 10^3) mod(1) \rfloor \times 2^{32}$
    $R \leftarrow X \| Y$
    $S \leftarrow S \parallel R$
    $i \leftarrow i + 1$
    $j \leftarrow j + 1$
  end if
end while
return S

4 Security analysis

In this section, we first exhibit a study of the security of our generator against attacks through a formal security analysis[8], key space and key sensitivity. Second, we perform the random analysis of the generator output by entropy, Histogram, NIST tests and Dieharder tests.

4.1 Formal security analysis

A formal security analysis of pseudo-random number generator guarantees that the generator is secure when the outputs are indistinguishable in polynomial time from another uniform distribution generated by another source [2], [10], [11].

Two ensembles X_n and Y_n are statistically close if their statistical difference is negligible. In that case they are also indistinguishable in polynomial time. However, the converse is untrue [10].

The statistical difference is given by the function :

$$\delta(n) = \frac{1}{2}\sum_{\alpha} \mid Pr[X_n = \alpha] - Pr[Y_n = \alpha] \mid$$

Now, we calculate the statistical differences between ten random sequences generated by our generator and ten others provided using Matlab's "rand" function.

Table 1 shows the results.

Table 1: Statistical Difference

Sequences	$\delta(n)$
1	0.000333370079837857
2	0.000333444103124018
3	0.000333406138545570
4	0.000333368516303501
5	0.000333278273508269
6	0.000333225315793545
7	0.000333399362138856
8	0.000333432623581588
9	0.000333268516504502
10	0.000333249517312451

Table 1 indicates that the statistical difference between the sequence from our PRNG and the one from Matlab is less than 0.034% for the ten comparaisons we made. We can say these differences are negligible, so our PRNG's sequences and Matlab's are indistinguishable.

4.2 Histogram

To measure the distribution of our generator's outputs, we use a visual test (i.e. the histogram) [13].

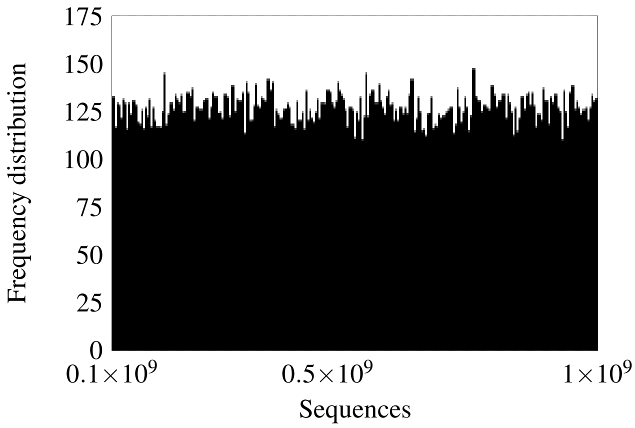

Fig. 3: Histogram test result for sequences in $[10^8; 10^9]$.

Figure 3 illustrates that the distribution of our generator's outputs is uniform.

4.3 Entropy

The Shannon entropy is another numerical test that evaluates the average amount of randomness contained in a given source [12].

Existence of all values should be equiprobable [13]. This test can be applied using the following formula :

$$H(S) = -\sum_{i=0}^{n} Pr(s_i) \times log_2 Pr(s_i)$$

Where $H(S)$ is the entropy value for the sequence, and $Pr(s_i)$ is the probability of the occurrence of each value.

We calculated the entropies of ten sequences generated by our PRNG and ten others generated by the "rand" function from Matlab. Table 2 shows the results.

Table 2: Comparison between the entropy of our generator and that of "rand" function from Matlab.

Sequences	Entropy of our PRNG	Entropy of rand function
1	26.040986434642150	25.690318043116390
2	26.070531376416571	26.026246156715211
3	26.011563791576325	26.182989117450119
4	25.817920777077662	25.858197136427929
5	26.085410174333433	26.142902729186530
6	25.989467468612918	26.082532219488901
7	25.931863811492090	25.965797296929013
8	26.054270536575130	25.992902729186530
9	26.038555943091972	25.984663540502476
10	26.146863978363647	26.178282303679637

The table shows that the sequences from our generator have entropies close to the entropies of those from Matlab's "rand" function.

4.4 Key Space

The size of key space is a prominent criterion of a cryptosystem. A large security key space makes exhaustive attacks impossible. The proposed generator is initialized by a key of any size more than four characters: $(size > 32bits)$.

Any key space of size larger than 2^{128} is computationally secured against exhaustive attacks [14].

Using **algorithm** 1, we calculate seven initial values $r, l_1, l_2, m_1, m_2, \theta_1$ and θ_2, of 64 bits each, i.e. 448 bits from the binary string of the key. Therefore, our space of initial values is 2^{448}, which is large enough to avoid any exhaustive attack as mentioned before.

4.5 Key sensitivity

The key sensitivity means that the smallest change in the secret key produces a big change in the pseudo-random sequence. This characteristic is central to make a highly secured generator against statistical and differential attacks [3], [7].

Our generator is based on a double pendulum which is a chaotic system. This makes it very sensitive to the initial conditions. Thus, any small difference between the keys produces very different outputs.

To examine the security of our generator we used several keys K_i with one bit of difference between each of them and the first key k_0, and we calculated the hamming distances between every output S_i and S_0.

The number $DH(S_i,S_j) = card\{e/x_e \neq y_e\}$, with $S_i = x_1x_2...x_N$ and $S_j = y_1y_2...y_N$, is the value of the hamming distance between two binary sequences.
This distance is given by:

$$DH(S_i,S_j) = \sum_{k=1}^{N} (x_k \bigoplus y_k)$$

The fact that a generator is very sensitive to the key makes the hamming distance vary in the neighborhood of $N/2$,(when N is the length of the sequences) resulting in the $DH(S_i,S_j)/N$ being about 0.50 for each pair of sequences.

In the next step, the aim is to generate a set of sequences S_i from the keys $\{k_i\}_{0\leq i\leq 48}$.

Let's consider k_0 = "*CHOKRI*" whose binary representation in ASCII code is $k_0 = (01000011\,01001000\,01001111\,01001011\,01010010\,01001001)_2$. The other 48 keys $\{k_i\}_{1\leq i\leq 48}$ are derived from k_0 through modifying only the i^{th} bit among the 48 bits of the k_0.

The result of the Hamming distance between the sequences is presented in Figure 4.

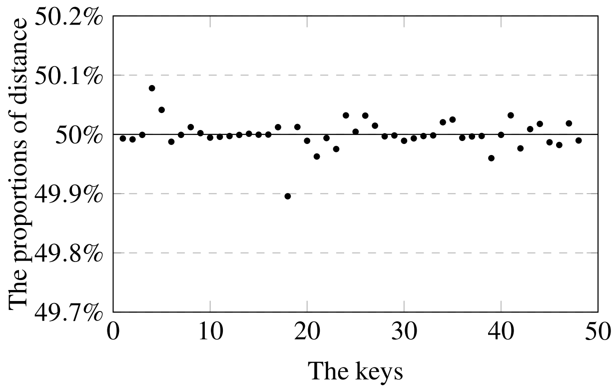

Fig. 4: The proportions of the Hamming distance.

Difference proportion between each sequence $S_i(1 \leq i \leq 48)$ and S_0 is about 50%, which implies that the proposed generator is purely sensitive to the initial conditions.

Sensitivity to key changes occurs due to the chaotic system that constructs the generator, suggesting that the generated sequences are chaotic and unpredictable.

A change of one bit between two keys leads to totally different initialization values and generated sequences.

4.6 Randomness level

NIST tests and DIEHARDER tests are used to measure the level of randomness of the bits sequences generated by our PRNG.

For each statistical test, P_{value} is calculated from the bits sequence. It is compared to a predefined threshold $\alpha = 0.01$, which is also called significance level. If P_{value} is greater than α, the sequence is considered to be random, and passed the statistical test successfully. Otherwise, the sequence does not appear random.

4.6.1 NIST

NIST tests suite consists of 15 tests [15] developed to quantify and evaluate the degree of randomness of the sequences produced by the cryptographic generators.

To apply the NIST tests to our generator, various keys are used to produce 1000 sequences as a first step. The size of each sequence is 10^6 bits. Table 3 presents the test results.

Table 3: NIST Statistical test suite results for 1000 sequences of size 10^6 bits each, generated by the our generator

NIST statistical test	P_{value}	Pass rate
Frequency	0.896345	989/1000
Block-Frequency	0.014100	982/1000
Cumulative Sums	0.465077	990/1000
Runs	0.455937	985/1000
Longest Run	0.163513	995/1000
Rank	0.345650	984/1000
FFT	0.832561	986/1000
Non-Overlapping	0.507534	991/1000
Overlapping	0.420827	997/1000
Universal	0.986658	993/1000
Approximate Entropy	0.705466	988/1000
Random Excursions	0.608559	606/612
Random Excursions Variant	0.498447	607/612
Serial	0.407401	991/1000
Linear Complexity	0.174728	995/1000

Table 3 reveals that NIST tests suite is successful. The p_{value} of all tests is greater than the minimum rate (0.01).

The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately 980 for a sample size 1000 binary sequences. The minimum pass rate for the random excursion (variant) test is approximately 598 for a sample size 612 binary sequences.

4.6.2 DIEHARDER

DIEHARDER tests consist of a set of statistical tests that measure the quality of randomness developed by George Marsaglia [16].

For DIEHARDER tests, 1000 sequences of 10^6 bits are generated by the proposed pseudo random numbers generator. The results are presented in Table 4.

Table 4 shows that DIEHARDER P_{values} are in the acceptable range of $[0,1)$, and all tests are successful.

Table 4: DIEHARDER Statistical test suite results for 1000 sequences of size 10^6 bits each, generated by the our generator

DIEHARDER test name	P_{value}	Assessment
Birthday	0.75887849	passed
Overlapping 5-permutation	0.90706791	passed
Binary rank (32 x 32)	0.68830507	passed
Binary rank (6 x 8)	0.99066355	passed
Bitstream	0.96862068	passed
OPSO	0.61437395	passed
OQSO	0.35136051	passed
DNA	0.96343430	passed
Stream count-the-ones	0.71485215	passed
Byte count-the-ones	0.61749066	passed
Parking lot	0.61013501	passed
2D circle	0.77047741	passed
3D spheres	0.28540625	passed
Squeeze	0.58848691	passed
Sums	0.11428089	passed
Runs	0.96215077	passed
Craps	0.87203328	passed
Marsaglia and Tsang GCD	0.70839018	passed
STS Monobit	0.96164295	passed
STS Runs	0.22671732	passed
STS Serial Test (Generalized)	0.65230427	passed
RGB Bit Distribution	0.80597401	passed
RGB Min Distance	0.34340374	passed
RGB Permutations	0.08477149	passed
RGB Lagged Sum	0.52026957	passed
RGB Kolmogorov-Smirnov	0.67656475	passed

5 Conclusion

In this paper, a new pseudo-random number generator based on a Double Pendulum was presented.

The generator was selected after a rigorous analysis that showed high dimensional chaos, which helped the generator produce more complex and unpredictable chaotic sequences.

The results of the statistical analyses, namely formal security analysis, histogram, entropy, key space, key sensitivity and randomness indicated that the proposed generator provided high security, which made it appropriate for practical encryption.

In the future paper, we will apply our proposed generator to image and audio encryption using a new method based on xor and permutation operations.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this article.

References

[1] F. Michael, D. David, A Fast Chaos-Based Pseudo-Random Bit Generator Using Binary 64 Floating-Point Arithmetic, Informatica **38**, 115-124 (2014)
[2] O. Reyad, Z. Kotulski, On Pseudo-Random Number Generators Using Elliptic Curves and Chaotic Systems, Applied Mathematics and Information Sciences, **9**, 1, 31-38 (2015)
[3] K. Charif, A. Drissi, and Z.A. Guennoun, A pseudo random number generator based on chaotic billiards, International Journal of Network Security, **19**, 479-486, (2017)
[4] T. Shinbrot, C. Grebogi, J. Wisdom, and J. Yorke, Chaos in a double pendulum, American Journal of Physiology, **60**, 491-499 - (1992)
[5] T. Yaren, S. Kizir, A.B. Yildiz, The motion characteristics of the double-pendulum system with skew walls, Mathematical Methods in the Applied Sciences, **42**, 475-487 (2019)
[6] M. Lampart, J. Zapomel, Electrical Equivalent Circuit Based Analysis of Double Pendulum System, in Proc. ICEEE 258-262 (2019)
[7] C. Nouar and Z.A. Guennoun, A Pseudo Random Bit Generator Based on a Modified Chaotic Map, International Journal of Network Security, **21**, 402-408, (2019)
[8] S. Ruhault, Security Analysis for Pseudo-Random Number Generators, Ph.D. thesis, Ecole normale superieure, Paris (2015)
[9] D Lambic, M Nikolic, Pseudo-random number generator based on discrete-space chaotic map, Nonlinear Dynamics, **90**, 223-232 (2017)
[10] O. Goldreich, Foundations of Cryptography : Basic Tools, Cambridge University Press, Cambridge, 103-118, (2004)
[11] L. Marangio, C. Guyeux, Entropy and Security of Pseudorandom Number Generators based on Chaotic Iterations, in Proc. ICETE, 402-407 (2019)
[12] C Li, D Lin, B Feng, J Lu, F Hao, Cryptanalysis of a Chaotic Image Encryption Algorithm Based on Information Entropy, IEEE Access, **6**, 75834-75842 (2018); J.-S. Zhang,A.-X. Chen, M. Abdel-Aty, Two atoms in dissipative cavities in dispersive limit: Entanglement sudden death and long-lived entanglement, J. Phys. B: Atom. Mol. Opt. Phy., **43** 025501 (2010).
[13] O. Salhab, N. Jweihan, M.A. Jodeh, M.A. Taha, and M. Farajallah, Survey paper pseudo random number generator and security tests, Journal of Theoretical and Applied Information Technology, **96**, 1951-1970. (2018)
[14] E. Hato and D. Shihab, Lorenz and rossler chaotic system for speech signal encryption, International Journal of Computer Applications, **128**, 25-33, (2015)
[15] NIST, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, SP 800-22, (2010)
[16] Robert G. Brown, Dieharder : A Random Number Test Suite, Duke University, Physics Department, 27708-0305, (2020)

Zine El Abidine Guennoun He is a full professor of Department of Mathematics at the Faculty of Science, Mohamed V University in Rabat, Morocco. He received his Ph.D. (1989). His research interests include non linear analysis, fixed point theory, differential equation, financial mathematics and cryptography.

Chokri Nouar He is received his Master's degree in mathematics and statistics, option cryptography and information security from Mohammed-V University in Rabat, Morocco. He is actually a PhD student in the Laboratory of Mathematics, statistic and applications. His major research interests include information security and cryptography.